CSC3320 System Level Programming
Lab Assignment 9 - Post-Lab

Due at 11:59 pm on Sunday, March 21, 2021

Purpose: Learn how to use array in C. Understand the basic memory address in C.

Part 1:

Write a C program named as getMostFreqChar.c that finds the most frequent letter from the input via ignoring the case sensitive and prints out its frequency. For example, sample outputs could be like below

$cat test.txt
This is a list of courses.
CSC 1010 - COMPUTERS & APPLICATIONS

$./getMostFreqChar test.txt
The most frequent letter is 's'. It appeared 8 times. Run the C program,

attach a screenshot of the output in the answer sheet.

```
[mpatel185@gsuad.gsu.edu@snowball ~]$ vi getMostFreqChar.c
[mpatel185@gsuad.gsu.edu@snowball ~]$ [mpatel185@gsuad.gsu.edu@snowball ~]$ gcc -o getMostFreqChar getMostFreqChar.c
[mpatel185@gsuad.gsu.edu@snowball ~]$ ./getMostFreqChar
Enter string: This is a list of courses. CSC 1010 - COMPUTERS & APPLICATIONS
The Most frequent letter is 's'. It appeared 8 times.
[mpatel185@gsuad.gsu.edu@snowball ~]$
```

Part 2:

When a variable is stored in memory, it is associated with an address. To obtain the address of a variable, the & operator can be used. For example, &a gets the memory address of variable a. Let's try some examples.
Write a C program addressOfScalar.c by inserting the code below in the main function.

Questions:
1) Run the C program, attach a screenshot of the output in the answer sheet.

```
[mpatel185@gsuad.gsu.edu@snowball ~]$ vi addressOfScaler.c
[mpatel185@gsuad.gsu.edu@snowball ~]$ gcc -o addressOfScaler addressOfScaler.c
[mpatel185@gsuad.gsu.edu@snowball ~]$ ./addressOfScaler
address of charvar = 0x7ffce9f05cdf
address of charvar - 1 = 0x7ffce9f05cde
address of charvar + 1 = 0x7ffce9f05ce0
address of intvar = 0x7ffce9f05cd8
address of intvar - 1 = 0x7ffce9f05cd4
address of intvar + 1 = 0x7ffce9f05cdc
[mpatel185@gsuad.gsu.edu@snowball ~]$
```

2) Attach the source code in the answer sheet

```
#include <stdio.h>


int main(){
    //initialize a char variable, print its address and the next
address
    char charvar = '\0';
    printf("address of charvar = %p\n", (void *)(&charvar));
    printf("address of charvar - 1 = %p\n", (void *)(&charvar - 1));
    printf("address of charvar + 1 = %p\n", (void *)(&charvar + 1));


    // intialize an int variable, print its address and the next
address
    int intvar = 1;
    printf("address of intvar = %p\n", (void *)(&intvar));
    printf("address of intvar - 1 = %p\n", (void *)(&intvar - 1));
    printf("address of intvar + 1 = %p\n", (void *)(&intvar + 1));
}
```

3) Then explain why the address after intvar is incremented by 4 bytes instead of 1 byte.

```
1 // intialize a char variable, print its address and the next address 2 char charvar = '\0';
3 printf("address of charvar = %p\n", (void *)(&charvar));
4 printf("address of charvar - 1 = %p\n", (void *)(&charvar - 1)); 5 printf("address of charvar +
1 = %p\n", (void *)(&charvar + 1));6
7 // intialize an int variable, print its address and the next address 8 int intvar = 1;
9 printf("address of intvar = %p\n", (void *)(&intvar));
10 printf("address of intvar - 1 = %p\n", (void *)(&intvar - 1)); 11 printf("address of intvar +
1 = %p\n", (void *)(&intvar + 1))
```

intvar is declared as an integer data type and it uses up to 4 bytes, not 1 byte to store the data

Part 3:

Write a C program addressOfArray.c by inserting the code below in the main function.

```
1 // initialize an array of ints
2 int numbers[5] = {1,2,3,4,5};
3 int i = 0;
4
5 // print the address of the array variable
6 printf("numbers = %p\n", numbers);
7
8 // print addresses of each array index
9 do {
10 printf("numbers[%u] = %p\n", i, (void *)(&numbers[i]));
11 i++;
12 } while(i < 5);

    // print the size of the array
    printf("sizeof(numbers) = %lu\n", sizeof(numbers));
```

Questions:

1) Run the C program, attach a screenshot of the output in the answer sheet.

```
[mpatel185@gsuad.gsu.edu@snowball ~]$ [mpatel185@gsuad.gsu.edu@snowball ~]$ vi addressOfArray.c
[mpatel185@gsuad.gsu.edu@snowball ~]$ [mpatel185@gsuad.gsu.edu@snowball ~]$ gcc -o addressOfArray addressOfArray.c
[mpatel185@gsuad.gsu.edu@snowball ~]$ ./addressOfArray
numbers = 0x7fff80262670
numbers[0] = 0x7fff80262670
numbers[1] = 0x7fff80262674
numbers[2] = 0x7fff80262678
numbers[3] = 0x7fff8026267c
numbers[4] = 0x7fff80262680
sizeof(numbers) = 20
[mpatel185@gsuad.gsu.edu@snowball ~]$
```

2) Check the address of the array and the address of the first element in the array. Are they the same?

Both address of array and first element of that array are the same

3) Write down the statement to print out the length of the array by using sizeof

operator.

```
printf("sizeof(numbers) = %lu\n", sizeof(numbers));
```

Submission:

- Upload an electronic copy (pdf) of your answer sheet to the folder named "Lab 9" in Google Classroom
- Please add the lab assignment number and your name at the top of your answer sheet.
- Upload the C files getMostFreqChar.c, addressOfArray.c and addressOfScalar.c to the folder named named "Lab 9" in Google Classroom
- Name your file in the format of Lab9_ FirstnameLastname (e.g Lab9_FilRondel.pdf)